IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **David Shippy**	§	Group Art Unit: **2181**
	§	
Serial No. **10/821,025**	§	Examiner: **Lai, Vincent**
	§	
Filed: **April 8, 2004**	§	Customer No.: **50170**
	§	
For: **Architected Register File System Utilizes Status and Control Registers to Control Read/Write Operations Between Threads**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

APPELLANT'S BRIEF (37 C.F.R. § 41.37)

This Appeal Brief is in furtherance of the Notice of Appeal filed February 2, 2007 (37 C.F.R. § 41.31).

The fees required under § 41.20(b)(2), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying Transmittal of Appeal Brief.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals and interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

The status of the claims involved in this proceeding is as follows:

1. Claims canceled: 10-15
2. Claims withdrawing from consideration but not canceled: NONE
3. Claims pending: 1-9 and 16-21
4. Claims allowed: NONE
5. Claims rejected: 1-9 and 16-21

The claims on appeal are: 1-9 and 16-21

STATUS OF AMENDMENTS

No amendments to the application were filed subsequent to mailing of the Final Office Action.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1:

The presently claimed invention provides an architected register file system at least configured to utilize a plurality of threads. See specification, page 1, lines 5-7; page 3, lines 15-17; page 9, line 25, to page 10, line 17, for example. The architected register file system comprises a plurality of register files. See 222 and 224 in FIG. 2, for example. Each register file of the plurality of register files at least corresponds to a respective thread of the plurality of threads. See specification, page 3, lines 17-20; page 8, line 25, to page 9, line 5, for example. The architected register file system comprises a plurality of status and control registers (SCR). See 218 and 220 in FIG. 2, for example. Each status and control register corresponds to a respective thread of the plurality of threads. See specification, page 3, lines 20-22; page 10, lines 7-9, for example. The architected register file system further comprises a plurality of control bit sets. See 256 and 258 in FIG. 2. Each control bit set corresponds to one status and control register. See specification, page 3, lines 22-24; page 10, lines 9-11, for example. Each control bit set is at least configured to allow a thread associated with an associated status and control register to utilize other register files associated with other threads. See specification, page 3, line 24, to page 4, line 2; page 10, line 11, to page 11, line 10; page 15, lines 1-20, for example.

Independent claim 7:

The presently claimed invention provides a method for utilizing a plurality of register files with associated status and control registers in a multithread system. See specification, page 1, lines 5-7; page 3, lines 15-17; page 9, line 25, to page 10, line 17, for example. Each register file of the plurality of register files at least corresponds to a respective thread of the plurality of threads. See specification, page 3, lines 17-20; page 8, line 25, to page 9, line 5, for example. The method comprises receiving an instruction for a first thread of the plurality of threads, the first thread being associated with a first status and control register. See specification, page 14, lines 21-22; step 302 in FIG. 3, for example. The method comprises decoding the instruction to at least determine performance operations. See specification, page 12, lines 2-11; page 14, lines 22-25; step 304 in FIG. 3, for example. The method further comprises determining if the first thread is enabled to at least utilize register files associated with other threads. See specification,

page 10, lines 11-15; page 12, lines 13-17; page 15, lines 1-20; step 306 in FIG. 3, for example. The method comprises executing the instruction utilizing at least one register file associated with a second thread of the plurality of threads. See specification, page 13, line 23, to page 14, line 8; page 15, lines 10-20; steps 312, 314, and 316 in FIG. 3, for example.

***Independent claim 16*:**

The presently claimed invention provides a method for utilizing a plurality of register files in a multithread system. See specification, page 1, lines 5-7; page 3, lines 15-17; page 9, line 25, to page 10, line 17, for example. The method comprises receiving an instruction for a first thread having an operations code field, a write field, and one or more read fields. Each operations code field defines a desired operation for the instruction. The write field defines an address location to which a result of the operation is to be stored. The at least one read field defines an address location from which data is to be read for the operation. See specification, page 9, lines 6-24; instruction 202 in FIG. 2, for example. The method comprises decoding the instruction. See specification, page 12, lines 2-11; page 14, lines 22-25; step 304 in FIG. 3, for example. The method comprises setting a first status and control register associated with the first thread and a second status control register associated with a second thread based on the decoding of the instruction. See specification, page 9, line 25, to page 12, line 21, for example. A first register file is associated with the first thread and a second register file is associated with the second thread. See specification, page 3, lines 17-20; page 8, line 25, to page 9, line 5, for example. The method further comprises determining whether the first thread is permitted to utilize the second register file associated with second thread based on at least one of the first status and control register or the second status and control register. See specification, page 10, lines 11-15; page 12, lines 13-17; page 15, lines 1-20; step 306 in FIG. 3, for example. The method comprises if the first thread is permitted to utilize the second register file, performing the operation utilizing the second register file by writing to or reading from the second register file associated with the second thread. See specification, page 13, line 23, to page 14, line 8; page 15, lines 10-20; steps 312, 314, and 316 in FIG. 3, for example.

***Independent claim 19*:**

The presently claimed invention provides an apparatus for utilizing a plurality of register files in a multithread system. See specification, page 1, lines 5-7; page 3, lines 15-17; page 9, line 25, to page 10, line 17, for example. The apparatus comprises a decoder that receives and decodes an instruction for a first thread having an operations code field, a write field, and one or more read fields. The operations code field defines a desired operation for the instruction. The write field defines an address location to which a result of the operation is to be stored. The at least one read field defines an address location from which data is to be read for the operation. See specification, page 9, lines 6-24; page 12, lines 2-11; page 14, lines 22-25; instruction 202 and decoder 216 in FIG. 2, for example. The apparatus comprises a first register file associated with the first thread and a second register file associated with the second thread. See specification, page 3, lines 17-20; page 8, line 25, to page 9, line 5; 222 and 224 in FIG. 2, for example. The apparatus comprises a first status and control register associated with the first thread and a second status and control register associated with the second thread. See specification, page 3, lines 20-22; page 10, lines 7-9; 218 and 220 in FIG. 2, for example. The decoder sets the first status and control register and the second status and control register based on the decoding of the instruction. See specification, page 9, line 25, to page 12, line 21, for example. The apparatus further comprises an address control that enables or disables access to the first register file and the second register file based on the first status and control register and the second status and control register. The address control enables the first thread to access the second register file if the first thread is permitted to utilize the second register file. See specification, page 10, lines 11-15; page 12, lines 13-17; page 15, lines 1-20; 254 in FIG. 2, for example. The apparatus comprises one or more execution units that perform the operation utilizing the second register file by writing to or reading from the second register file associated with the second thread based on the address control. See specification, page 13, line 23, to page 14, line 8; page 15, lines 10-20; 226 in FIG. 2, for example.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection on appeal are as follows:

I. Claims 1-5, 7, 8, and 16-21 are rejected under 35 U.S.C. § 102(b) as being anticipated by *Sollars* (U.S. Patent No. 5,900,025);

II. Claims 6 and 9 are rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over *Sollars*.

ARGUMENT

I. 35 U.S.C. § 102, Alleged Anticipation of Claims 1-5, 7, 8, and 16-21

The Office rejects claims 1-5, 7, 8, and 16-21 under 35 U.S.C. § 102(b) as allegedly being anticipated by *Sollars* (U.S. Patent No. 5,900,025). Appellant respectfully traverses this rejection.

IA. 35 U.S.C. § 102, Alleged Anticipation of Claims 1-5, 7, and 8

Sollars teaches a processor having a hierarchical control register file. *Sollars* teaches a primary control register file that comprises a plurality of control registers that are organized into control register sets. See *Sollars*, col. 5, lines 19-43. *Sollars* teaches that control register sets are dynamically allocated to contexts and threads during operation. See *Sollars*, col. 5, lines 43-46. *Sollars* states:

> Except for the initial allocations at system start up time, the control register sets at the various levels are dynamically allocated to the contexts and the threads during operation. Besides modifications as a result of instruction execution, i.e., status update and the like, the control registers are directly accessible and modifiable using instructions from the standard instruction sets of exemplary processor 10. A context and thread based privilege structure is employed to control the direct accesses and modifications.

Sollars, col. 5, lines 43-53. An auxiliary control register file augments the primary control register file. See *Sollars*, col. 5, lines 55-66.

Sollars also states:

> Under the presently preferred embodiment, a thread is initially

> conferred a standard thread privilege, which allows the thread to access and modify its own set of control registers. On an as needed basis, one of the threads of each context can be temporarily conferred a context privilege, which allows the context privileged thread to also access and modify its context's context level control register set as well as its peer threads' control register sets. Similarly, on an as need basis, one of the contexts (more specifically, one of the context privileged threads) can be temporarily conferred a system privilege, which allows the system privileged context to also access and modify the system level control register set as well as any one of the other "lower" level control register sets.

Sollars, col. 3, lines 45-58. See also *Sollars*, col. 15, lines 60-66. Thus, *Sollars* teaches a hierarchical control register structure where threads have associated thread privileges. A thread may access certain context levels within the control register structure based on the thread's privilege.

Sollars also teaches primary and secondary operand register files. With respect to the primary operand register file, *Sollars* states:

> Primary operand register file 22a includes a number of registers for performing the conventional function of storing instruction operands in a new and innovative manner. Preferably, primary operand register file 22a is a scalable uni/multidimensional as well as virtually/physically addressable register file, used for storing integer as well as floating point operands, as disclosed in copending U.S. patent application, application Ser. No. 08/401,411, having common inventorship with the present invention, which is hereby fully incorporated by reference.

Sollars, col. 5, lines 21-30. The Office Action alleges that the primary and secondary control register files of *Sollars* are equivalent to the plurality of status and control registers and that the primary and secondary operand register files of *Sollars* correspond to the plurality of register files in claim 1. *Sollars* does not teach that each operand register file corresponds to a thread. Rather, *Sollars* teaches that the primary operand register file includes a number of registers for performing the function of storing instruction operands. More specifically, *Sollars* teaches that the primary operand register file is a scalable uni/multidimensional as well as a virtually/ physically addressable register file, used for storing integer as well as floating point operands. However, *Sollars* does not teach that each operand register file corresponds to a respective thread.

The Final Office Action states:

> Cited in the action is column 2, lines 2-6, which says, "At the time the third highest level of the logical hierarchy is a number of control register sets for controlling concurrent execution of multiple peer process threads (hereinafter simply threads) for each of the concurrently executing peer contexts." There is a mention of hierarchy, but hierarchy is simply the manner in which the register files are organized. Sollar goes on to say, "in one implementation of the preferred embodiment, eight sets of context control registers are provided for concurrently supporting up to eight active contexts, and 64 sets of thread control registers are provided for concurrently supporting up to eight active threads for each of the active contexts" (Column 2, lines 7-12). Thus control registers do indeed correspond to threads and not merely a privilege level. Given this, the register file that contains the control registers must indeed also correspond to a thread.

Thus, the Final Office Action argues that 64 sets of thread control registers support up to eight active threads and then somehow concludes that each of the 64 sets of thread control registers corresponds to a respective one of the threads. However, this passage is referring to the **control registers** and not the **operand registers**. The Office made this distinction, but fails to apply the distinction when addressing this particular feature of the claim. The Final Office Action switches back and forth between control registers and operand registers when it allegedly supports the rejection. However, clearly *Sollars* does not teach or suggest that each **operand** register file of the plurality of register files at least corresponds to at least one thread of the plurality of threads.

Still further, *Sollars* does not teach or suggest a plurality of control bit sets that are configured to allow a thread associated with an associated status and control register to utilize other register files associated with other threads. The Office Action alleges that *Sollars* teaches this function at col. 15, lines 60-66, which states:

> On an as needed basis, one of the threads of each context is temporarily conferred a context privilege, which further allows the context privileged thread to access and modify the thread's context level control register set 104 as well as the peer threads' control register sets 106.

The above portion appears to teach that a thread may be allowed to access and modify peer threads' control register sets. However, the Office Action alleges that the primary and secondary **operand** register files are equivalent to the plurality of register files of claim 1. Thus, the Office Action has not shown that *Sollars* teaches of fairly suggests a plurality of control bit sets that are configured to allow a thread to utilize other **operand** register files associated with other threads. In fact, *Sollars* does not even teach that each operand register file corresponds to a thread;

therefore, *Sollars* cannot teach or suggest the feature of allowing one thread to utilize the register file of another thread, as recited in claim 1.

The Final Office Action states:

> Sollars goes on to discuss the conditions in which threads may be allowed to access and modify a peer thread's control register sets (See column 16, lines 5-15). In the discussion, Sollars mentions that certain execution exceptions may necessitate such actions to be taken. These exceptions are stored within the register file (See figure 9E and column 13, lines 25-28: the SWFLAG register contains control bits for traps) and thus there are indeed a plurality of control bit sets that are configured to allow a thread to utilize other operand register files associated with other threads.

Appellant respectfully disagrees. FIG. 9E of *Sollars* does illustrate a **control register** with control bits. With respect to the control register shown in FIG. 9E, *Sollars* states:

> HWFLAG and SWFLAG control registers 112e-112f are used to store the current values of various hardware defined and software defined flags for the thread. SWFLAG control register 112f is also used to facilitate the exchange of status and commands with a child MLR of the thread.

Sollars, col. 8, lines 58-63. The cited portion of *Sollars* states:

> As shown in FIG. 13, a MLR stored in WCSF 24 is invoked and given control in like manner as interrupts and exceptions, through interrupt/exception buffer 40 of exemplary processor 10. Under the presently preferred embodiment, the invoking thread passes a number of parameters to the invoked MLR through the invoking macro-trap, including the offset required to locate the SWFLAG array entry which is used to provide the address of the first control register of the partitioned subset 107 of auxiliary control register file 20b allocated to controlling execution of the invoked MLR, and for exchanging status and commands between the invoking thread and the invoked MLR. The located SWFLAG array entry is automatically synchronized with the status/command control register 113d of the allocated partitioned subset. The interrupt/exception controller of processor 10 services the buffered macro-trap if there is at least one partitioned subset of auxiliary control register file 20b available. For a more detailed description of MLRs, their invocation and usage, see the incorporated by reference copending U.S. Patent application.

Col. 13, lines 25-44. Neither this portion nor any other portion of *Sollars* teaches a set of control bits that allows a thread to utilize other **operand** register files associated with other threads. The Examiner appears to reference arbitrary portions of the reference without proffering any analysis

as to why the portions somehow teach the claim features. Therefore, the Final Office Action fails to establish a *prima facie* case of anticipation.

The applied reference fails to teach or fairly suggest each and every claim feature; therefore, *Sollars* does not anticipate claim 1. Independent claim 7 recites features addressed above with respect to claim 1 and is allowable for similar reasons. Since claims 2-5 and 8 depend from claims 1 and 7, the same distinctions between *Sollars* and claims 1 and 7 apply for these claims. Additionally, claims 2-5 and 8 recite further combinations of features not suggested by the reference.

Therefore, Appellant respectfully requests withdrawal of the rejection of claims 1-5, 7, and 8 under 35 U.S.C. § 102(b).

IB. 35 U.S.C. § 102, Alleged Anticipation of Claims 16-21

Independent claims 16 and 19 are allowable for at least the reasons stated above with respect to claims 1-5, 7, and 8. More specifically, with respect to claim 16, the Final Office Action alleges that *Sollars* teaches setting a first status and control register associated with a first thread and a second status and control register associated with a second thread based on the decoding of the instruction, wherein a first register file is associated with the first thread and a second register file is associated with the second thread. The Final Office Action does not indicate where *Sollars* allegedly teaches these features. Certainly, many microprocessors in the prior art decode instructions and set status and control registers accordingly. However, *Sollars* does not teach or suggest a first register file being associated with a first thread and a second register file being associated with a second thread, as shown above with respect to independent claim 1.

The Final Office Action alleges that *Sollars* teaches determining whether the first thread is permitted to utilize the second register file associated with the second thread based on at least one of the first status and control register or the second status and control register at col. 16, lines 5-15, which states:

> The conditions under which the privileges are dynamically conferred to and withdrawn from the various threads and contexts are implementation dependent. Under the presently preferred embodiment, the conditions include, in particular, the conditions of encountering certain execution exceptions, under which a thread

> will be temporarily conferred the privilege for accessing and modifying its context's control register set 104, and a context will be temporarily conferred the privilege for accessing and modifying the system set of control registers 102, by the exception service routines, while the exceptions are being serviced.

This portion of *Sollars* does appear to teach temporarily conferring a privilege for accessing and modifying a set of **control** registers. However, this portion fails to teach or fairly suggest determining whether a first thread is permitted to utilize a second register file associated with a second thread and writing to or reading from the second register file according to an operation of an instruction.

Once again, the Examiner makes a distinctions that the control registers of *Sollars* are equivalent to the claimed control registers and that the operand register files of *Sollars* are equivalent to the claimed register files, but then ignores these distinctions when applying the teachings of *Sollars* to the specific features of the claims. Claim 16, like claim 1, does not recite determining whether a first thread has permission to access the **control** registers of a second thread; rather, the claims recite determining whether a first thread has permission to access an **operand register file** associated with a second thread. However, the Final Office Action repeatedly refers to portions of *Sollars* that teach a first thread accessing its context's control registers.

The Final Office Action alleges that *Sollars* teaches performing the operation of the instruction utilizing the second register file by writing to or reading from the second register file associated with the second thread at col. 13, lines 25-38, which is reproduced above. This cited portion appears to teach a SWFLAG control register. However, the cited portion of *Sollars* in no way teaches performing the operation of an instruction by writing to or reading from an operand register file associated with a different thread, as alleged in the Final Office Action.

The applied reference fails to teach or fairly suggest each and every claim feature; therefore, *Sollars* does not anticipate claim 16. Independent claim 19 recites features addressed above with respect to claim 16 and is allowable for similar reasons. Since claims 17, 18, 20, and 21 depend from claims 16 and 19, the same distinctions between *Sollars* and claims 16 and 19 apply for these claims. Additionally, claims 17, 18, 20, and 21 recite further combinations of features not suggested by the reference.

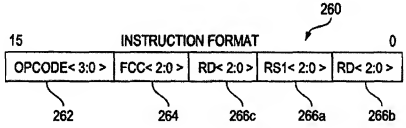
Therefore, Appellant respectfully requests withdrawal of the rejection of claims 16-21 under 35 U.S.C. § 102(b).

IB(1). 35 U.S.C. § 102, Alleged Anticipation of Claim 17

With respect to claim 17, the Final Office Action alleges that *Sollars* teaches reading data from the second register file based on an address in a read field within the one or more read fields at col. 15, lines 60-66, which is reproduced above. Again, the cited portion appears to teach that a thread may be allowed to access and modify peer threads' control register sets. However, this is very different from **reading data based on an address in a read field of an instruction**. The Final Office Action appears to reference a seemingly arbitrary portion of the reference without proffering any analysis as to why the portion somehow teaches the claim feature. Therefore, the Final Office Action fails to establish a *prima facie* case of anticipation for claim 17.

IB(2). 35 U.S.C. § 102, Alleged Anticipation of Claim 18

With respect to claim 18, the Final Office Action alleges that *Sollars* teaches writing a result of the operation to the second register file based on an address in a write field at FIG. 16A, which is as follows:



The cited portion appears to teach an instruction format. However, this is very different from **writing a result to a register file associated with a different thread based on an address in a write field of an instruction**. The Final Office Action appears to reference a seemingly arbitrary portion of the reference without proffering any analysis as to why the portion somehow teaches the claim feature. Therefore, the Final Office Action fails to establish a *prima facie* case of anticipation for claim 18.

II. 35 U.S.C. § 103(a), Alleged Obviousness of Claims 6 and 9

The Office rejects claims 6 and 9 under 35 U.S.C. § 103(a) as allegedly being unpatentable over *Sollars*. This rejection is respectfully traversed.

With respect to claims 6 and 9, the Office Action alleges that *Sollars* teaches one bit of a doublet that is configured to correspond to a read function and one bit of the doublet being configured to correspond to a write function, where the bits of the doublet enable or disable reading to or writing from a register file associated with a different thread, at col. 10, lines 51-67. The cited portion of *Sollars* states:

> As shown in FIG. 9a, TCTL control register 112c is used to store a number of control values for controlling the execution of the thread. For the illustrated embodiment, TCTL control register 112c comprises a number of flag values (AFE, MFE, SFE, and LFE) denoting whether update of the HWFLAG control register 112e is to be enabled/disabled for various instructions (i.e. Add, Multiply and Accumulate, Shift and Logic). Additionally, TCTL control register 112c comprises an sl value denoting whether the destination cache line of a store operation is to be locked, an sc value denoting whether the store data of a store operation is cacheable, and an sb value denoting whether store operations are to be forced to complete in program order. TCTL control register 112c further comprises an ll value denoting whether the destination cache line of a load operation is to be locked, an lc value denoting whether the load data of a load operation is cacheable, and an lb value denoting whether load operations are to be forced to complete in program order. Lastly, TCTL control register 112c comprises a clk_div value for halting execution of the particular thread.

Sollars, col. 10, lines 45-64. Appellant respectfully disagrees. As stated above, *Sollars* does not teach or suggest that each operand register file corresponds to a thread. *Sollars* also fails to teach or suggest a plurality of control bit sets that are configured to allow a thread associated with an associated status and control register to utilize other register files associated with other threads. *Sollars* does appear to teach a TCTL control register that includes flags for enabling various functions, such as locking a cache line, setting whether a store or load operation is cacheable, or setting whether a store or load operation is forced to complete in program order. However, the Office Action proffers no analysis as to why this is somehow equivalent to separately enabling a thread to read from a register file associated with a different thread or separately enabling a thread to write to a register file associated with a different thread.

The Final Office Action states:

> Please refer to arguments presented above. The TCTCL is in control of the various control registers, including the SWFLAG, which does allow for access other register files, as explained above.

Appellant respectfully disagrees. The SWFLAG is used to facilitate the exchange of status and commands with a child MLR of the thread. However, the Final Office Action fails to show that *Sollars* teaches or suggests a plurality of control bit sets that are configured to allow a thread associated with an associated status and control register to utilize other **operand** register files associated with other threads. The Final Office Action does not explain why the TCTCL and/or the SWFLAG is somehow equivalent to the claimed feature.

Appellant asserts that *Sollars* simply fails to teach or suggest such a feature. Given a lack of any suggestion in the cited prior art to provide a bit doublet in a status and control register to separately enable a thread to read from or write to a register file associated with a different thread, a person of ordinary skill in the art would not have found it obvious to modify the prior art to include this feature. The mere fact that a prior art reference can be readily modified does not make the modification obvious unless the prior art suggested the desirability of the modification. *In re Laskowski*, 871 F.2d 115, 10 U.S.P.Q.2d 1397 (Fed. Cir. 1989) and also *see In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992) and *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1993). The Office may not merely state that the modification would have been obvious to one of ordinary skill in the art without pointing out in the prior art a suggestion of the desirability of the proposed modification.

The Office Action has not shown that the prior art teaches or suggests a bit doublet in a status and control register for separately enabling a thread to write to a register file associated with a different thread. Therefore, the Office Action does not establish a *prima facie* case of obviousness. If the Patent Office does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985).

Therefore, Appellant respectfully requests withdrawal of the rejection of claims 6 and 9 under 35 U.S.C. § 103(a).

CONCLUSION

In view of the above, Appellant respectfully submits that claims 1-9 and 16-21 of the present application are not taught or suggested by the applied prior art. Accordingly, Appellant requests that the Board of Patent Appeals and Interferences overturn the rejections set forth in the Final Office Action.

Respectfully submitted,

Stephen R. Tkacs
Reg. No. 41,534
Walder Intellectual Property Law, P.C.
P.O. Box 832745
Richardson, TX 75083
(214) 722-6422
AGENT FOR APPELLANT

CLAIMS APPENDIX

1. An architected register file system at least configured to utilize a plurality of threads, comprising:

a plurality of register files, wherein each register file of the plurality of register files at least corresponds to a respective thread of the plurality of threads;

a plurality of Status and Control Registers (SCR), wherein each SCR corresponds to a respective thread of the plurality of threads; and

a plurality of control bit sets, wherein each control bit set corresponds to at least one SCR, and wherein each control bit set is at least configured to allow a thread associated with an associated SCR to utilize other register files associated with other threads.

2. The architected register file system of claim 1, wherein the architected register file system further comprises a decoder, wherein the decoder at least determines desired operations for an instruction.

3. The architected register file system of claim 1, wherein plurality of control bits further comprise a plurality of bit doublets, wherein a first bit of a bit doublet corresponds to a read function, and wherein a second bit of the bit doublet corresponds to a write function.

4. The architected register file system of claim 3, wherein the architected register file system further comprises:

an address control, wherein the address control at least determines addresses with the

plurality of register files; and

at least one execution unit, wherein the execution is at least configured to perform the operations of a input instruction within the plurality of register files.

5. The architected register file system of claim 3, wherein the plurality of bit doublets further comprises that each bit doublet at least corresponds to enabling the use of at least one register file associated with another thread.

6. The architected register file system of claim 5, wherein each bit doublet of the plurality of bit doublets further comprises:

at least one bit is at least configured to correspond to a read function, wherein a logic high or `1` enables the first thread to read from another register file; and

at least one bit is at least configure to correspond to a write function, wherein a logic high or `1` enables the first thread to write to another register.

7. A method for utilizing a plurality of register files with associated SCRs in a multithread system, wherein each register file is at least associated with one thread of a plurality of threads, comprising:

receiving an instruction for a first thread of the plurality of threads, wherein the first thread is at least associated with a first SCR;

decoding the instruction to at least determine performance operations;

determining if the first thread is enabled to at least utilize register files associated with other threads; and

executing the instruction, wherein the step of executing utilizes at least one register file associated with a second thread of the plurality of threads.

8. The method of claim 7, wherein the step of determining if the first thread is enabled, further comprises measuring logical levels of control bits associated with the first SCR, wherein the control bits comprise a plurality of bit doublets, and wherein each bit doublet at least corresponds to enabling the use of at least one register file associated with another thread.

9. The method of claim 8, wherein the step of measuring further comprises determining if any bits are `1` or logic high, wherein the `1` or the logic high enables the first thread to read or write to another register file.

16. A method for utilizing a plurality of register files in a multithread system, the method comprising:

receiving an instruction for a first thread having an operations code field, a write field, and one or more read fields, wherein the operations code field defines a desired operation for the instruction, wherein the write field defines an address location to which a result of the operation is to be stored, and wherein the at least one read field defines an address location from which data is to be read for the operation;

decoding the instruction;

setting a first status and control register associated with the first thread and a second status and control register associated with a second thread based on the decoding of the instruction, wherein a first register file is associated with the first thread and a second register file

is associated with the second thread;

determining whether the first thread is permitted to utilize the second register file associated with the second thread based on at least one of the first status and control register or the second status and control register; and

if the first thread is permitted to utilize the second register file, performing the operation utilizing the second register file by writing to or reading from the second register file associated with the second thread.

17. The method of claim 16, wherein performing the operation comprises:

reading data from the second register file based on an address in a read field within the one or more read fields.

18. The method of claim 16, wherein performing the operation comprises:

writing a result of the operation to the second register file based on an address in the write field.

19. An apparatus for utilizing a plurality of register files in a multithread system, the apparatus comprising:

a decoder that receives and decodes an instruction for a first thread having an operations code field, a write field, and one or more read fields, wherein the operations code field defines a desired operation for the instruction, wherein the write field defines an address location to which a result of the operation is to be stored, and wherein the at least one read field defines an address location from which data is to be read for the operation;

a first status and control register associated with the first thread;

a first register file associated with the first thread;

a second status and control register associated with a second thread;

a second register file associated with the second thread, wherein the decoder sets the first status and control register and the second status and control register based on the decoding of the instruction, wherein a first register file is associated with the first thread and a second register file is associated with the second thread;

an address control that enables or disables access to the first register file and the second register file based on the first status and control register and the second status and control register, wherein the address control enables the first thread to access the second register file if the first thread is permitted to utilize the second register file; and

one or more execution units that perform the operation utilizing the second register file by writing to or reading from the second register file associated with the second thread based on the address control.

20. The apparatus of claim 19, wherein the first status and control register comprises a first control field that indicates whether to enable reading from the second register file.

21. The apparatus of claim 20, wherein the first status and control register comprises a second control field that indicates whether to enable writing to the second register file.

EVIDENCE APPENDIX

NONE

RELATED PROCEEDINGS APPENDIX

NONE